9/494,911

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 15/40 — A1

(11) International Publication Number: WO 91/12580

(43) International Publication Date: 22 August 1991 (22.08.91)

(21) International Application Number: PCT/GB91/00200

(22) International Filing Date: 8 February 1991 (08.02.91)

(30) Priority data:

9002876.2	8 February 1990 (08.02.90)	GB
9002874.7	8 February 1990 (08.02.90)	GB
9100082.8	3 January 1991 (03.01.91)	GB

(71) Applicant *(for all designated States except US)*: HEWLETT-PACKARD COMPANY [US/US]; 300 Hanover Street, Palo Alto, CA 94304 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)* : WHITTAKER, Steven [GB/GB]; HARRISON, Keith [GB/GB]; Hewlett-Packard Limited, Filton Road, Stoke Gifford, Bristol BS12 6QZ (GB).

(74) Agent: SMITH, Martin, Stanley; Stevens Hewlett & Perkins, 1 St. Augustine's Place, Bristol BS1 4UD (GB).

(81) Designated States: AT (European patent), BE (European patent), CH (European patent), DE (European patent), DK (European patent), ES (European patent), FR (European patent), GB (European patent), GR (European patent), IT (European patent), LU (European patent), NL (European patent), SE (European patent), US.

Published

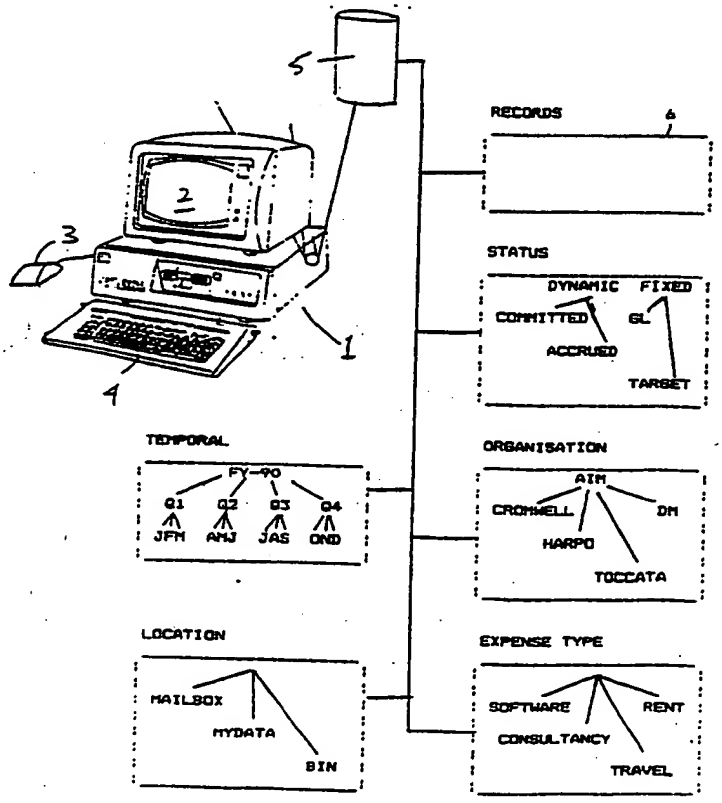
With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD AND APPARATUS FOR GRAPHICAL INTERROGATION OF A DATABASE

(57) Abstract

A graphical method of interrogating a computer database is provided, the database having a number of records and a number of dimensions in which each record is represented, the dimensions including headings and the method consisting in providing a blank elementary select table, adding user-determined row and column headings and executing a program which fills in the cells with data corresponding to the set intersections of the row and column headings.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	ES	Spain	MG	Madagascar
AU	Australia	FI	Finland	ML	Mali
BB	Barbados	FR	France	MN	Mongolia
BE	Belgium	GA	Gabon	MR	Mauritania
BF	Burkina Faso	GB	United Kingdom	MW	Malawi
BG	Bulgaria	GN	Guinea	NL	Netherlands
BJ	Benin	GR	Greece	NO	Norway
BR	Brazil	HU	Hungary	PL	Poland
CA	Canada	IT	Italy	RO	Romania
CF	Central African Republic	JP	Japan	SD	Sudan
CG	Congo	KP	Democratic People's Republic of Korea	SE	Sweden
CH	Switzerland			SN	Senegal
CI	Côte d'Ivoire	KR	Republic of Korea	SU	Soviet Union
CM	Cameroon	LI	Liechtenstein	TD	Chad
CS	Czechoslovakia	LK	Sri Lanka	TG	Togo
DE	Germany	LU	Luxembourg	US	United States of America
DK	Denmark	MC	Monaco		

Method and Apparatus for Graphical Interrogation of a Database

The invention relates to a method and apparatus for graphical interrogation of a database. The implementation of mouse-driven computer applications in a windows environment has materially simplified the general use of computers. Facility with a keyboard and knowledge of computer languages are no longer necessary. These benefits are felt particularly at managerial level and here it is important to be able to derive database information for monitoring and decision making purposes. Formulation of queries is an important managerial function.

Database interrogation depends heavily on the logical formulation of queries and special text-based languages have been developed as a consequence. An alternative graphical system of interrogation is desirable. A graphical interrogation technique is described in the paper "Query by Example", Proceedings of the National Computer Conference 1975 by M.M. Zloof. Here queries are formulated by filling in a table corresponding to a given relation in a relational database. Fixed and exemplary elements are distinguished in the table and a further output table is derived which displays the intersection or the union of the sets entered. This system suffers from two major disadvantages. Firstly, the tabular format of the queries is dictated by the established relation tables of the database and secondly the formulation of the query requires expertise in logic.

An object of the present invention is to provide a simple to use free format query facility in graphical form.

According to one aspect of the invention there is provided a method of interrogation of a computer database, the database having a number of records and a number of dimensions in which each record is represented, the dimensions including headings and the method consisting in providing a blank elementary select table, adding user-

determined row and column headings for the table from those available in the dimensions, thereby defining rows and columns and empty cells at the crossings of the rows and columns, and executing a program which fills in the cells with data corresponding to the set intersections of the row and column headings.

Preferably there are provided further steps of adding one or more further user determined row and/or column headings for the table from those available in the dimensions, and re-executing the program to fill in the empty cells.

In a preferred embodiment of the invention the records are records of quantities and the said data corresponding to the set intersections in the summation of the quantities of the records of the intersection.

Preferably the dimensions include sub-headings which may be used as column and row sub-headings in the select table. It is convenient to arrange that the headings and sub-headings for the rows and columns may be stacked to qualify the sets for which intersections are sought.

The dimensions may be tree structures having nodes representing the said headings from which branch further nodes representing the said sub-headings, each record including a locator field designating its node in each dimension and for each node there being available a list of the records appropriate at that node.

It will be appreciated that a free form query structure as proposed by the present invention may require the simultaneous calculation of a very large number of intersections, far larger, in fact, than the number allowed by the more restricted graphical or structured language techniques employed hitherto. A preferred programming technique makes this possible in an acceptable processing time, thereby rendering the system practical for large databases. This technique provides that lists of node

records are stored as bit strings of length corresponding to the total number of records and with bits turned on or off according to whether the record for the bit number is present at the node or not and the determination of set intersections is effected by successive logical AND operations on the bit strings for the nodes selected in the select table. With this arrangement advantage can be taken of the very fast bit string comparison techniques built in to most computer languages.

Although it is possible to store bit strings for each node of each dimension this is not necessary. Preferably bit strings are stored for only the leaf nodes, namely those at the ends of the branches. For processing queries at higher levels a transient bit string for a higher level node is obtained by an OR operation (union) on the bit strings of the leaf nodes which depend from it.

There are some query statements that do not readily lend themselves to being included in the free form tabular structure described above without imposing processing time handicaps. These include NOT operators (exclusion qualifiers) and size qualifiers such as "greater than" or "less than". Preferably, a textual language facility is added whereby a query answer table which has been derived by the free form technique can be qualified.

According to another aspect of the invention there is provided apparatus for performing the above described method, the apparatus comprising a computer with a storage medium holding a database, the database having a number of records and a number of dimensions in which each record is represented, the dimensions including headings, and the computer being programmed to present a blank elementary select table whereby a user may add row and column headings for the table from those available in the dimensions, thereby defining rows and columns and empty cells at the crossings of the rows and columns, the program being further effective to fill in the cells with data

corresponding to the set intersections of the row and column headings.

Another aspect of the invention provides for the use of natural language input. In order to make computer information systems more widely accessible and more 'user friendly', research effort is being expended to design computer systems that can interpret queries posed in 'natural language' i.e. language as ordinarily used by humans as opposed to a specialised database query language such as SQL.

It is known to use natural language input in combination with other forms of input to a computer system in a complementary manner. Examples are discussed in a joint paper entitled "Synergistic Use of Direct Manipulation and Natural Language" from the Artificial Intelligence Centers of SRI International and Lockheed Missiles and Space Co.

According to another aspect of the invention there is provided in a system as described above where input means are provided to permit a user to specify information selection criteria, further input means for permitting a user to make natural language input and further comprising means for combining the attributes selected using the first input means with those embodied in the natural language input and which is operable to interpret the natural language input in the context of the other input.

The invention will further be described with reference to the accompanying drawings, of which:-

Figure 1 is a schematic view of apparatus in accordance with the invention comprising a programmed computer with a database;

Figure 2 is a diagram of the screen presentation for implementing the method of the invention with the apparatus of Figure 1;

Figure 3 is a multi-part diagram where at (a) to (f) are shown successive stages in the building of the

query table of Figure 2;

Figure 4 is a multi-part diagram illustrating the extension of a free format query table in accordance with the invention;

Figure 5 is a diagram illustrating the contents of a record in the database of the apparatus of Figure 1;

Figure 6 is a diagram illustrating the bit string comparison technique employed in the preferred method in accordance with the invention;

Figures 7 and 8 are further examples of query tables;

Figure 9 is a flow chart showing processing of an NL query; and

Figure 10 is a multi-part diagram illustrating a tree structure of the system.

Referring to Figure 1 there is shown a general purpose computer conveniently of the personal computer (PC) kind and preferably part of a network of similar machines, although this is not illustrated. The computer 1 has a screen 2, and a mouse 3, a keyboard 4 and a hard disk 5. The computer runs under a Windows environment and the mouse is used in conventional manner to select items from pull-down menus and by reference to icons.

A database is stored on the hard disk 5. In this illustration the database is for the domain of financial planning over a given time period in an organisation which has several projects running and which incurs expenses of several kinds. Thus, the database has a set 6 of records, each of which has a record number, an expense value and fields which give location and other information. A more detailed description of a record will be given later.

In addition, the database stored on the hard disk 5 has five dimensions into which each record falls. Each domain has a tree structure with nodes and branches. The five dimensions are: STATUS; ORGANISATION; EXPENSE TYPE; TEMPORAL; and LOCATION.

STATUS DIMENSION

The status dimension has two primary nodes; DYNAMIC and FIXED. The DYNAMIC node branches to two further nodes: COMMITTED and ACCRUED. The FIXED node branches to nodes TARGET and GL. Each node represents a status of an expense (record). The status of an expense will change as it passes through the system.

ORGANISATION DIMENSION

The organisation dimension has a node AIM which branches to project nodes CROMWELL; TOCCATA; HARPO and DM. These nodes represent projects to which the expenses are assigned.

EXPENSE TYPE DIMENSION

The TYPE node branches to SOFTWARE; CONSULTANCY; RENT and TRAVEL. These are the types of the expenses.

TEMPORAL DIMENSION

This is a calendar dimension having a node FY90 (Financial Year 1990) which branches to "quarter" nodes Q1; Q2; Q3 and Q4. The quarter nodes branch to respective "month" nodes: J; F; M; etc.

LOCATION DIMENSION

The LOCATION node branches to system location nodes to where the expense has been assigned: MAILBOX; MYDATA; and BIN.

Referring now to Figure 2 there is shown a screen presentation typical in the execution of the invention. The screen has multiple windows. At the left-hand side is a set of windows representing the dimensions described above: window 7 for the STATUS dimension; window 8 for the ORGANISATION dimension; window 9 for the EXPENSE TYPE dimension; window 10 for the TEMPORAL dimension and window 11 for the LOCATION dimension.

At the right hand side is a window 12 which shows a typical expense record with the name TRAVEL 1. At the top part of the record are shown the nodes of the dimension which the record occupies, namely TEMPORAL; JAN 90; STATUS;

COMMITTED; ORGANISATION; HARPO; EXPENSE TYPE, TRAVEL and LOCATION, MYDATE. This information is stored in the record on the database. Other fields in the record gives the illustrated details of the expense and its value.

At the centre of the screen is a window 13 which has a free form query select table. The table in this illustration has been set up to interrogate the database by filling in column headings and sub-headings with LOCATION, MYDATA; TEMPORAL, Q1-09; STATUS; COMMITTED, ACCRUED and TARGET and by filling in the row headings and sub-headings with ORGANISATION, AIM, CROMWELL, TOCCATA, HARPO and DM. The table created is thus a 3 x 4 table with twelve cells at the crossings of the rows and columns. By clicking the mouse on EVALUATE, the program is triggered to fill in the cells with the totals of the values of the records which are the intersections of the sets dictated by the table.

The column and row headings and sub-headings are selected from the node names of the dimensions by means of clicking and dragging with the mouse. Any combination can be selected and it is immaterial whether the headings are in rows or columns. From the illustrated column headings it will be seen that headings may be stacked - e.g. "Mydata"; "Q1-09" and the three status headings. The program evaluates the intersection of the sets thus defined and adds the values of the expenses to display them in the cells.

If the database were essentially a non-numerial database then the cell would display the number of records in the intersection. The records could be listed by expanding the cell into another window or by scrolling in the cell, for example.

In Figure 3 there is illustrated the successive stages in building the query table 13 of Figure 2. Initially, the user is presented with an elementary select table 14, Figure 3(a). By clicking and dragging with the mouse to the column query 25 the user inserts the "Mydata"

heading from the LOCATION dimension, Figure 3(b). Then he inserts the heading "Q1-09" from the TEMPORAL dimension, Figure 3(c). Next he selects sub-headings "Committed", "Accrued" and "Target" from the STATUS dimension as sub-headings, Figure 3(d). These column headings are thus stacked and already an intersection set of records corresponding to the logical AND of the column headings has been established in the computer memory in a manner to be described.

Figures 3(e) and 3(f) show the successive stages of selecting the row heading "AIM" and sub-headings "Cromwell", "Toccata", "Harpo" and "DM". When a select table has been used it is stored so that it may be re-used or modified to give a new select table.

It should be appreciated that, after Figure 3 (c), the row heading "AIM" could have been used instead as another column heading prior to the insertion of the status headings in Figure 3 (d) so that there would be no row headings in the select table, as shown in Figure 3 (g). In practice it usually produces a more readable table to use both row and column headings but a user could use just row headings or just column headings if desired.

In Figure 4 there is illustrated the successive stages in building a query table. The starting point is taken as table (c) in Figure 3. Next the user selects the row heading "AIM" from the ORGANISATION dimension, Figure 4(a). By clicking on the "execute" button the user executes the program to evaluate the intersection of the sets. Here there is only one cell and one answer.

Having considered the answer, the user may conclude that he requires a break-down. He can achieve this by expanding the query select table and re-executing the evaluation program. By clicking on the column query mark 50 and the "committed" node in the displayed STATUS dimension, the user qualifies the column headings by the "committed" status. This is shown in Figure 4(b). Then, by clicking on

the AIM row heading 51 and the row query 52, the user automatically qualifies the AIM heading by the four project sub-headings "Cromwell"; "Toccatta"; "Harpo" and "DM". Further cells are automatically introduced at this stage and execution of the program fills all the cells in with their appropriate figures as shown in Figure 4(c).

Referring now to Figure 5 there is shown the format of a record. The record has a number according to its address in the record file. This is not explicitly stored in the record itself. At the beginning of the record is a text field 16 for record type, a number field 17 for record value, and then follow a locator field 18 and free fields 19. The locator field has five addresses corresponding to the nodes which the record occupies in the five dimensions. Thus, for the example of the record at 12 in Figure 2, the locator addresses would be for Q1-09 (TEMPORAL); Committed (STATUS); Harpo (ORGANISATION); Travel (EXPENSE TYPE) and Mydata (LOCATION). The free fields 19 carry the detailed information shown at 12 in Figure 2.

As the appropriate data is entered on the "record sheet" shown as 12 in Figure 2, so the dimension leaf nodes are updated. Each dimension leaf node carries a record of the record numbers appropriate to that node. This information is stored in a particularly useful way, as illustrated in Figure 6. Here there is a node list which is a field of successive bytes having a total number of bits corresponding to the total number of records in the database. If a particular record is appropriate to the node then the respective bit number of the node list bit string is turned on (status 1). If not it has status 0.

As successive headings and sub-headings are selected for the query table, the respective node bit strings are operated upon to produce further bit strings. Thus, if a higher level heading such as Q1 is selected, the appropriate leaf node bit strings (JFM) are ORed to produce

an intermediate union bit string. Any such intermediate bit strings are finally ANDed with each other and with leaf node bit strings as appropriate to produce further bit strings which represent the records belonging to the intersection of the node sets. This process is repeated for successive selections and it will be seen that in this way bit strings representative of the record sets for the cells of the table can be rapidly derived. It is then a simple matter to scan the values of the records for insertion of the totals in the cells.

As an illustration, Figure 6 shows a first node bit string 20 with records 1, 3, 6, 7 and 8 present; a second node bit string 21 with records 2, 3, 5, 6 and 9 present and the intersection bit string 22 of these with records 3 and 6 present.

In addition to deriving the intersection of bit string sets, the computer language allows the union of sets and other set operations to be executed. For example, NOT OR and XOR instructions can be executed. In order to facilitate the introduction of other instructions, including quantitative instructions such as "greater than", "equals", and "less than", the screen of Figure 2 has a natural language query box 24 whereby query text can be entered by keyboard. Conveniently such queries operate as the results of a free form query as displayed.

The free form query table may act as a "context" for a natural language query, and so influence how that NL query is interpreted by the system, and, in turn, influence the form of the response to the NL question.

In this augmentation of the approach described above it is possible for both the STRUCTURE and CONTENT of a table to act as a context for an NL query. For example, the entire table shown in Figure 7 can act as a context. Against this context, a query such as "Summarise the expenses for all quarters in FY-91" will yield the table in Figure 8. Here the query is assumed to range over the

restricted world of expenses for TRAVEL and for AIM. Crucially, however, the structure of the context table is preserved in the output table: the labels for AIM and TRAVEL remain.

By selecting a tabular context for an NL query, a user is specifying some parameters in advance of those expressed in the NL query. This approach can enhance user productivity by providing flexibility and obviating the need to type in long NL sentences.

Considering the query:

"Summarise the expenses for all quarters in FY91"

when posed in the tabular form shown in Figure 7 we will describe how such a table can be used to contextualise an NL query. First however, it is useful to consider the basic linguistic processing applied to a sentence, irrespective of whether it is given a tabular context.

Figure 9 shows the standard main steps in processing an NL query 45. Standard NL processing techniques are used at step 49 to derive from the sentence a semantic representation. At step 51, discourse processing and database translation techniques are used, along with any other relevant input 47, to derive an unambiguous query 53 which is then evaluated in the database at step 55 and the result 57 is provided to the user.

One possible implementation uses the Core Language Engine, developed at SRI International (Alshawi et al 1989), but other programs could be used.

For the above sentence, the system derives the following representation (here simplified and altered for the purposes of exposition):

(SHOW-SUMMARY
(Set X s.t. (EXPENSE-FORM X) & (FOR-TIME X Q)
(QUARTER Q) & IN-PERIOD(Q fy-91)))]

This can be read as follows: "show a summary of the set consisting of all those values X, where X is an expense for, X is for a time period Q, Q is a quarter, and Q is in

the time period FY-91. (Here, "s.t." means "such that"). To find the relevant expenses in the database, the expression which corresponds to the NOUN PHRASE "the expenses for quarters in FY-91",

(SET .....)

must be evaluated against the database. This is done by translating the (SET ...) expression into a formal query in the database query language, and running the query against the database.

(It is a simple matter to translate this (SET ...) representation into a database query in a database query language, and depending on the database query scheme used - eg. SQL or Prolog -- the final database query will look similar or different to the representation above).

The query is executed against the database to return the set of values X which satisfy the constraints placed on them. Here we are assuming that X is some representation of an expense form -- eg. if a unique number is used for every distinct expense form, the above expression would evaluate to a set such as (23,43,52,16,27,....) where each number denotes an expense-form.

We then apply the SHOW-SUMMARY procedure to this set; this looks up the expense form details associated with every expense-form identifier and represents them on the screen in the desired format.

The contextualisation of sentences requires augmentation of the process of interpreting the sentence. It has been found that for some applications, queries to the system typically include all of the relevant restrictive information in the main noun phrase of the sentence. For this reason, when considering tabular contexts, we shall describe inserting the tabular restrictions at the noun phrase level only. If we assume the tabular context of Figure 7 this introduces the contextual restricts on X, that:

(FOR ORGANISATION X aim) & (FOR-purpose X travel)

These restrictions can be derived from the table on the basis of a set of rules which -- for example -- translate the tabular label "travel" into the constant used to describe "travel" in the linguistic world. Knowledge built into the linguistic program enables it to infer that the appropriate restriction for travel is "FOR-PURPOSE", and similarly the predicate "FOR-ORGANISATION" for the other label.

This representation of the tabular context can be merged with that for the sentence, to produce the following representation for the contextualised sentence:

(SHOW-SUMMARY (SET (X s.t (EXPENSE-FORM X) & (FOR-TIME X Q)

(FOR-ORGANISATION X aim) & (FOR-PURPOSE X travel)
(QUARTER Q) & IN-PERIOD (Q fy-91)))]

Hence the evaluation of the contextualised sentence will produce a set of those values X, restricted as before in pure linguistic processing, and in addition restricted so that X is for the organisation AIM and X is for the purpose of travel.

In this way, when the query has been given a tabular context, then the attributes already present in this table are combined with those derived from the linguistic phrase to produce the new structure of the table. At the same time, when searching the database for all "expenses for all quarters in FY-91", we further restrict the search in line with the attributes specified in the table -- i.e. TRAVEL and AIM.

In addition, in the system described above, the response to a natural language (NL) question or command can be represented as a summary table, as well as a set of expense forms; the user can state which is preferred.

For example, the user can type "Summarise the committed expenses for all projects in FY-90". The response can be in the form of a table of exactly the same kind as those produced by standard tabular querying.

Such a table produced by the NL is not simply a static image displayed on the screen. It is a dynamic object which has all the properties of a table produced through the standard "point-and-click" route of tabular querying. Therefore such a table can be developed by further selection and formatting, from the graphical models, and it is possible to "drill-down" from any of its cells to retrieve the relevant expense forms. This is possible by effectively representing the content of the NL query in the "language" of summary tables so everything that the NL query expressed can be conveyed in the tabular query.

The present invention also has more general application in combining MENU-based querying with NL-based querying. Reverting to the above example of contextualising NL against tables. In the abstract, this involves contextualising against a set of discrete selections from a space of possibilities (e.g. select expense status = PLANNED, expense type = TRAVEL, and so on.) Such a space of possibilitites is very often represented as a MENU system, where the user is asked to select from a set of options presented to him/her. In that example, the domain of expense dimensions could be represented in menu terms as follows. When we enter the system, we can do one of five things:

A Specify some STATUS dimensions
Specify some ORGANISATION dimensions
Specify some EXPENSE TYPE dimensions
Specify some TIME dimensions
State a restriction in natural language

We choose the first option, and the system gives us these choices:

B I want FIXED expenditure
I want DYNAMIC expenditure
State a restriction in natural language

You choose "DYNAMIC" but because DYNAMIC has subcategories you are given another choice:

C I want all DYNAMIC expenditure
I want PLANNED expenditure
I want COMMITTED expenditure
I want ACCRUED expenditure
State a restriction in natural language

You make your choices, say PLANNED and COMMITTED, and hit a button to indicate that you have finished your selection on this screen. You return to the previous screen B, and again indicate that you have finished at this level. The menu then gives you these options:

A2 Specify some ORGANISATION dimensions
Specify some EXPENSE-TYPE dimensions
Specify some TIME dimensions
State a restriction in natural language

Let's say you do a similar thing in the expense-type menu "space" and when you have finished that you have chosen the following:

(I want PLANNED or COMMITTED expenditure,
I want TRAVEL expenditure)

At this point you have the following options:

A3 Specify some ORGANISATION dimensions
Specify some TIME dimensions
State a restriction in natural language

You decide on NL, and the system gives you this screen:

E Please enter NL query:
You type:
"for the first three quarters in FY-91"

At this point you return to the menu main screen, which will be the same as A3. Your restrictions are now effectively:

(I want PLANNED or COMMITTED expenditure,
I want TRAVEL expenditure, I want Q1 or Q2 or Q3 expenditure)

Let's say you have made all the selections you want. Your selections and NL can be translated into an expression for evaluation, such as:

```
(SET X s.t. (EXPENSE-FORM X) & (FOR-PURPOSE X travel)
        & [ (HAS-STATUS X PLANNED) OR (HAS-STATUS X
        COMMITTED)]
        & [ (FOR-TIME X q1) OR (FOR-TIME X q2) OR
        (for-time x Q3)])
```

When a request of menu selections to be evaluated against the database is made, the query evaluation phase will return all Xs that satisfy these constraints. Constraints that arise from selections WITHIN the same "dimensions" are ORed. Restrictions underlying the menu selections eg. "TRAVEL" can be converted into NL-orientated restrictions in a manner similar to that described for the tabular format example. That is, these restrictions can be derived from the menu selections on the basis of a set of rules which eg. translate the menu label TRAVEL into the constant used to describe "travel" in the linguistic world. Knowledge built into the linguistic program enables it to infer that the appropriate restriction for TRAVEL is "FOR-PURPOSE". Any restrictions from the same dimension are ORed.

It should be noted that rather than processing only complete sentences, a system according to the invention may be configured to process fragments of sentences.

The tree structures of Figure 1 (typically the TEMPORAL tree) represent dimensions, or in other words classifications, of the domain. There follows a discussion of the interpretation of the classification trees and the selector row and column headings which are combined to provide the database query mechanism.

The nodes of the classifications trees correspond, either directly or indirectly, to a set of data base records in the following way:

If the classification tree node is a leaf node, i.e. has no children, then it represents a set of records

obtained by retrieving all records from the database that have a particular value for a specified data field. Should this set of records be retrieved from the database then these records may be 'remembered' to avoid this computation in the future.

If the classification tree node is not a leaf node, i.e. it has children nodes, then it represents the union of the sets of data records corresponding to each of the children nodes.

Thus, it is possible to associate a database query with each node in the classification tree. For example, in a tree having the format of Figure 10(a) the leaf nodes d, e and f are associated with the queries.

d: key = "d value"

e: key = "e value"

f: key = "c value"

The two non-leaf nodes are associated with the queries

b: union of query (key = "d value") and query (key = "e value")

and

a: union of (union of query (key = "d value") and query (key = "e value"))
and query (key = "c value").

The query for a can be re-written as follows

a: union of query (key = "d value")
and query (key = "e value")
and query (key = "c value")

That is, each node in the classification trees corresponds to a union of one or more sets of data records, each corresponding to a database query of the form key = "value".

A selector is a two-dimensional report whose column and row headings are specified by the user by selecting nodes from the classification trees. Once the columns and rows have been specified, then the implied query may be

posed and the results displayed to the user.

The rows and columns are handled in the same way. Consider only the column headings. The column headings form a treee, the leaves of which correspond to each column. Each node in the headings tree is associated with a particular node in the classification trees. Each node in the headings tree is assumed to represent the set of records formed by intersecting the set of records associated with the node's parent and the set of records associated with the classification node. The root node, always labelled '?', is assumed to refer to the entire database.

Thus, if there is a headings tree of the form of Figure 10(b) then the heading of 'c' represents a set of records obtained by the query

intersection of (intersection of whole data base
and (query associated with classification node a))
and (query associated with classification node c)

rearranging gives

intersection of (whole database)
and (query associated with classification node a)
and (query associated with classification node c)

That is, each heading represents a conjunction of queries, each query being the disjunction of sets of records.

When the user requests that a 'selector' be 'executed', then a data structure representing each row and column is created. This data structure is an access plan. That is, it represents a set of activities that will be performed that results in a set of records being associated with each row and column. As described above, each column heading represents a conjunction of disjunctions of sets of records. That is, can be represented as a data structure with the following structure

intersect(union(query11, query12, query13, ...),

```
union(query21, ....),
.
.
)
```

This structure may be rewritten, and so optimised, by using a number of rules:

1. union(Whole database, Set) == Whole Database
2. intersection(Whole database, Set) == Set
3. intersection(Set, Set) == Set
4. union(Set, Set) == Set
5. intersection(Set given by key = valuel, Set given by key = Value2) == Empty Set if Valuel and Value2 are different.
6. intersection(Set, EmptySet) == Empty Set
7. union(Set, EmptySet) == Set

Once the structure has been optimised, then the query is posed. This results in sets of records being associated with each row and column in the selector. These sets of values are permanently associated with the row and column headings.

The set of records for each cell in the report can now be obtained by taking the intersection of the appropriate row and column sets.

There follows a description of the processing required to re-pose a query defined by a selector.

The user is permitted only to amend the row and column headings by adding additional nodes to existing row and column headings. If the existing query was posed, then the original leaf nodes would have been annotated with their access plan and the resulting set of records. This information may be used to optimise the query execution.

Assume the original columns tree has the format of Figure 10(b). This is then evaluated which causes the nodes c, d, e and f to be annotated with their query access plan and the resulting set of records. This tree is then amended to the form of Figure 10(c).

To generate the access plan for the new nodes g and h, it is necessary only to intersect the set of values associated with node c with the sets of values corresponding to classification nodes g and h. This eliminates the need to consider the whole columns tree when generating the access plans - and so increases speed.

There follows a description of the data structures of the system. There are a number of classification trees each corresponding to a different 'dimension' in the database. There is a data structure to store the roots of each of the trees:

classification_tree(tree_name, root_node_id).

Each node in the classification trees is given a unique identifier and has one of the two following structures:

classification_tree_node(unique_identifier,
'leafnode',
tree_name,
unique_identifier_of_parent,
query_associated_with_node,
'uncached' or set of records from database).
classification_tree_node(unique_identifier,
'non-leafnode',
tree_name,
unique_identifier_of_parent,
list_of_unique_identifiers_for_each_child).

Lastly, it is necessary to track which hierarchy nodes have been selected by the user. This will be used to modify a selector, and then to ensure that the herarchy nodes are redisplayed in their default state.

Each existing selector has a data structure that describes its current state. This has the format:

selector(unique_identifier,
column_structure,
row_structure,
set_of_currently_selected_cells,

set_of_currently_selected_rows,
set_of_currently_selected_columns)

As the user selects cells in the selector for further processing, the set of currently selected cells is maintained. Similarly for the set of currently selected rows and columns.

The rows and columns are defined as follows:

heading_root_node(RootNodeId).

heading_node(unique_identifier,
maps_to_classification_tree_node_id,
unique_identifier_of_parent,
list_of_unique_identifiers_for_children,
access_plan,
set_of_records)

where the access_plan is a tree structure whose nodes are defined as follows:

access_plan_node ::= intersection(list of access_plan nodes)
or union(list of access_plan_nodes)
or leaf_set(set of records)
or query(query)

If the access plan or the set_of_records has not yet been computed then an indicator value is used.

There follows a description of the computation that is performed in order to modify an existing selector to build a new selector.

To amend an existing selector the user performs the following actions:

1. Select a number of classification tree nodes.

This results in a set of 'selected classification nodes'.

2. Select one or more row and columns in the target selector.

This results in a set of 'currently selected rows' and 'currently selected columns'.

3. Select a 'Format' option.

This causes a copy to be made of the entire data structure that represents the selector. Because structure sharing is used this is not as expensive as it might appear. New row and/or column nodes are created, one for each classification node in the selected set. These nodes are linked into the set of children associated with each of the selected row and column nodes. The new selector is then displayed. All nodes marked by the user in the classification trees, row headings or column headings are reset back to their default state and the sets of selected items cleared.

In order to evaluate a selector the following steps are necessary:

1. Make a copy of the old selector to work upon.

2. Compute the set of records that are associated with each row and column.

3. Redraw the new selector, calculating the values for each cell. The cell value is a function of the set of records defined by the intersection of the records associated with the row and column the cell lies in.

With regard particularly to the second step, consider each leaf node of the row (and column) header tree. If the leaf node contains a set of previously computed records then no further processing is necessary - the old results may be re-used. If not previously computed, then the row header tree is traversed upwards towards the root building up an access plan. Traversing is stopped once either the root node is reached, or a node that contains a set of records (from a previous evaluation) is reached. A new access plan is generated and optimised using the rules mentioned in section 1.3. This access plan is then executed and the resulting set of results stored in the leaf node. In this way much of the computation performed in earlier selectors can be re-used.

CLAIMS

1. A method of interrogation of a computer database, the database having a number of records and a number of dimensions in which each record is represented, the dimensions including headings and the method consisting in providing a blank elementary select table, adding user-determined row and column headings for the table from those available in the dimensions, thereby defining rows and columns and empty cells at the crossings of the rows and columns, and executing a program which fills in the cells with data corresponding to the set intersections of the row and column headings.

2. A method as claimed in Claim 1 including the further steps of adding one or more further user-determined row and/or column headings for the table from those available in the dimensions, and re-executing the program to fill the empty cells.

3. A method as claimed in Claim 1 or Claim 2 wherein the records are records of quantities and the said data corresponding to the set intersections is the summation of the quantities of the records of the intersection.

4. A method as claimed in any of the preceding claims wherein the dimensions include sub-headings which may be used as column and row sub-headings in the select table.

5. A method as claimed in Claim 4 wherein the headings and sub-headings for the rows and columns may be stacked to qualify the sets for which intersections are sought.

6. A method as claimed in Claim 4 or Claim 5 wherein the dimensions are tree structures having nodes representing the said headings from which branch further nodes representing the said sub-headings, each record includes a locator field designating its node in each dimension and for each node there is available a list of

the records appropriate to that node.

7. A method as claimed in Claim 6 wherein lists of records at each node are stored as a bit strings of length corresponding to the total number of records and with bits turned on or off according to whether the record for the bit number is present at the node or not, and the determination of set intersections is effected by successive logical AND operations on the bit strings for the nodes selected in the select table.

8. A method as claimed in any of the preceding claims wherein the results of a query can be qualified by one or more textual language queries which may include exclusion qualifiers or size delimiters.

9. Apparatus for performing the method of any of Claims 1 to 8, the apparatus comprising a computer with a storage medium holding a database, the database having a number of records and a number of dimensions in which each record is represented, the dimensions including headings, and the computer being programmed to present a blank elementary select table whereby a user may add row and column headings for the table from those available in the dimensions, thereby defining rows and columns and empty cells at the crossings of the rows and columns, the program being further effective to fill in the cells with data corresponding to the set intersections of the row and column headings.

10. A computer system comprising apparatus as claimed in Claim 9 having first input means for permitting a user to specify said select table and second input means for permitting a user to make natural language input and further comprising means for combining the attributes selected using the first input means with those embodied in the natural language input and which is operable to interpret the natural language input in the context of the other input.

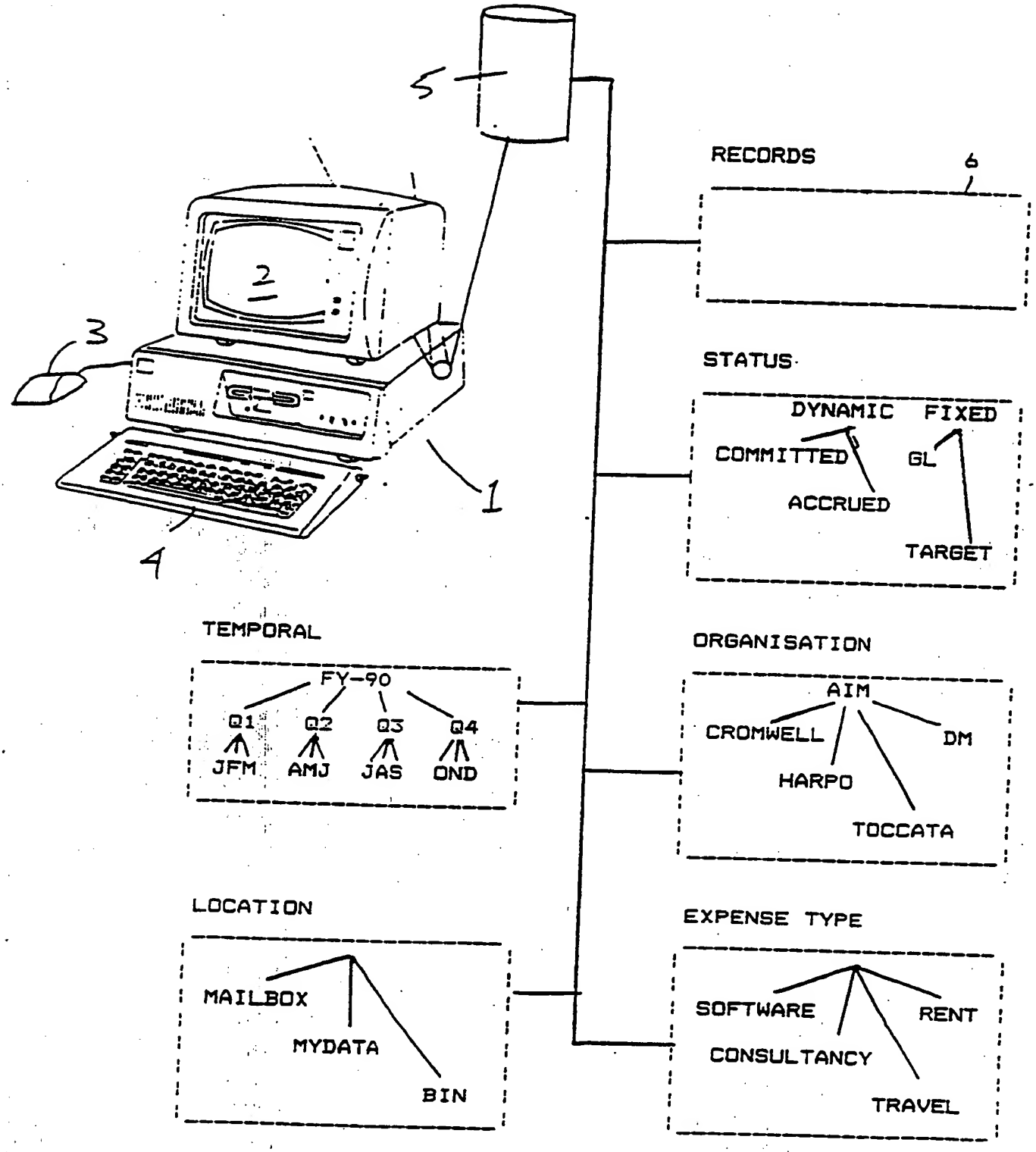

5
RECORDS
6
1
2
3
4
STATUS
DYNAMIC
FIXED
COMMITTED
GL
ACCRUED
TARGET
TEMPORAL
FY-90
Q1
Q2
Q3
Q4
JFM
AMJ
JAS
OND
ORGANISATION
AIM
CROMWELL
DM
HARPO
TOCCATA
LOCATION
MAILBOX
MYDATA
BIN
EXPENSE TYPE
SOFTWARE
RENT
CONSULTANCY
TRAVEL

FIG. 1

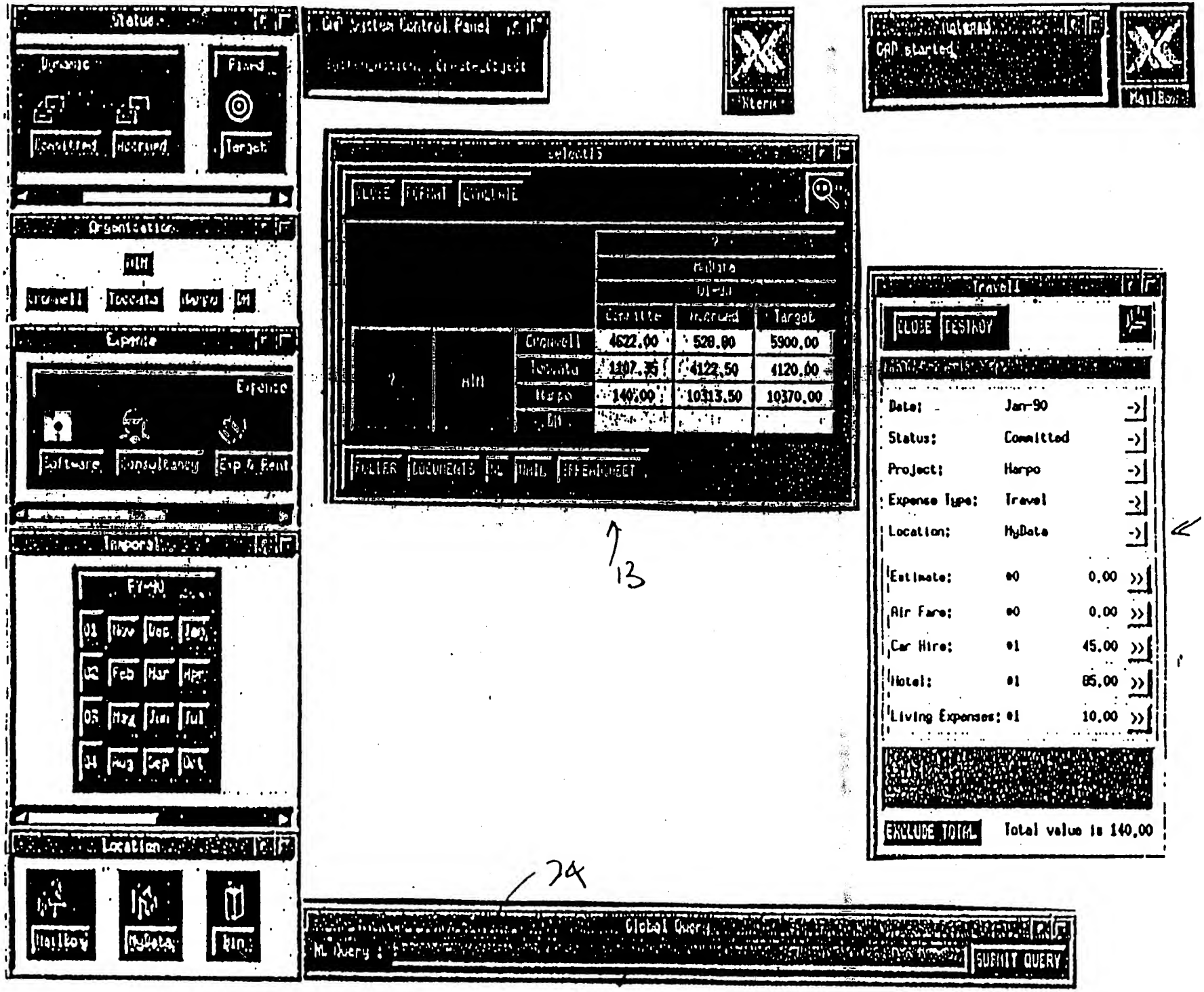

Status
Dynamic
Committed
Accrued
Fixed
Target
Organisation
HIH
Cromwell
Toccata
Harpo
BM
Expense
Expense
Software
Consultancy
Exp & Rent
Imports
FY-90
Jan
Feb
Mar
Jul
Aug
Sep
Oct
Location
Mailbox
MyData
Bin
Select5
CLOSE
FORMAT
CONSOLE
FILTER
DOCUMENTS
MAIL
Cromwell
Toccata
Harpo
BM
4622.00
528.00
5900.00
1207.35
4122.50
4120.00
140.00
10313.50
10370.00
13
Travel
CLOSE
DESTROY
Date:
Jan-90
Status:
Committed
Project:
Harpo
Expense Type:
Travel
Location:
MyData
12
Estimate:
0.00
Air Fare:
0.00
Car Hire:
45.00
Hotel:
85.00
Living Expenses:
10.00
EXCLUDE TOTAL
Total value is 140.00
14
Global Query
SUBMIT QUERY

FIG 2

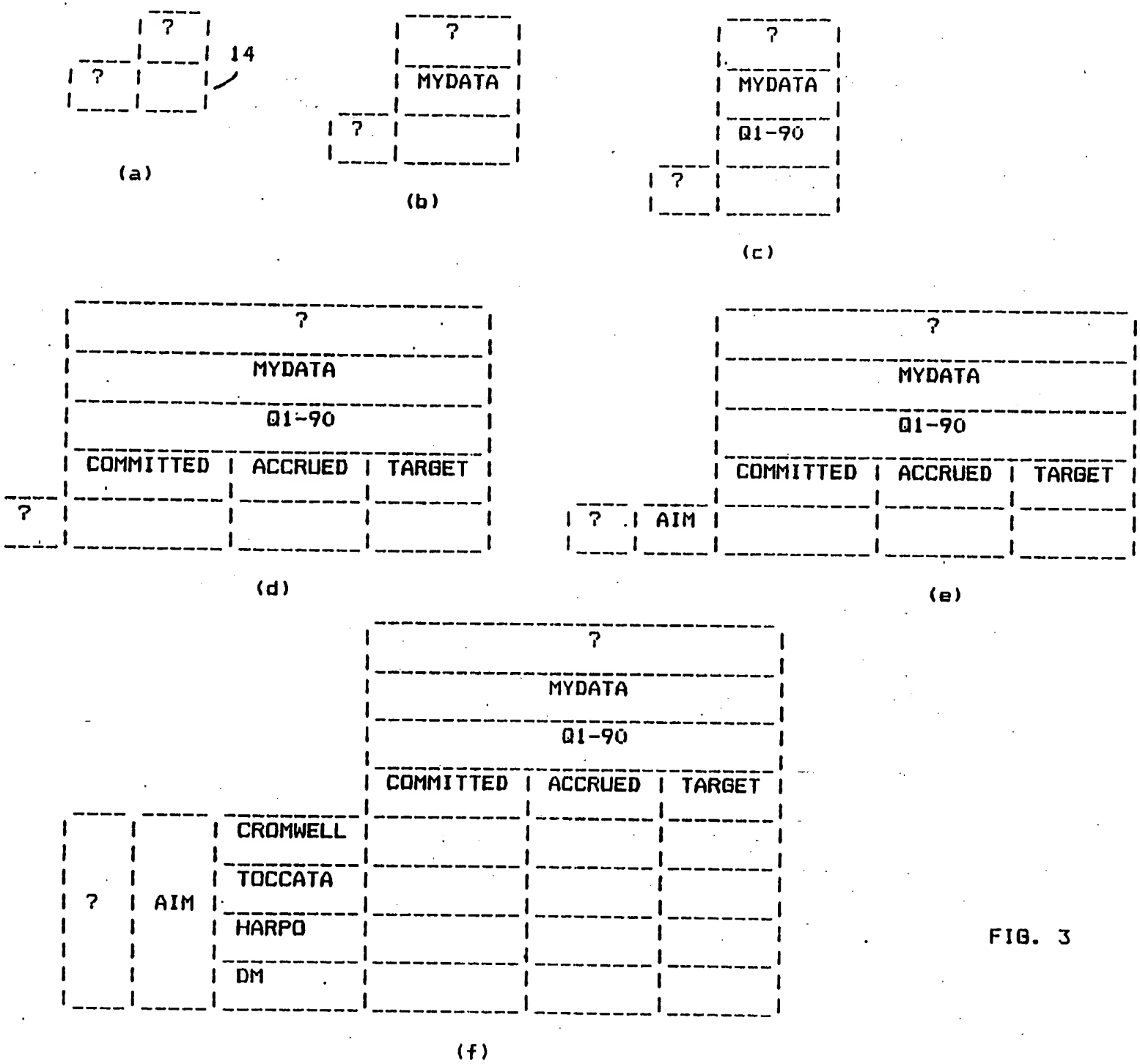
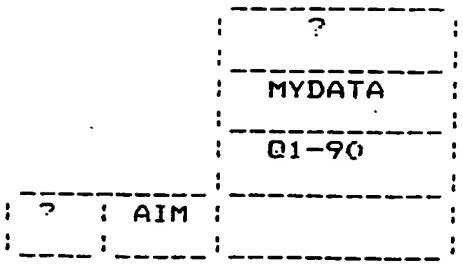
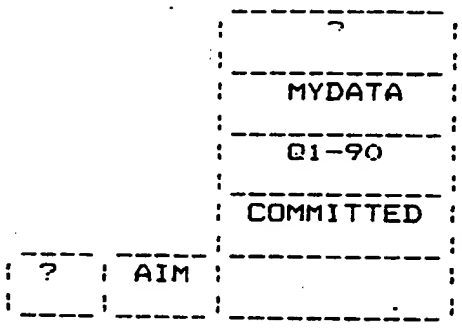

?
14
(a)
?
MYDATA
(b)
?
MYDATA
Q1-90
(c)
?
MYDATA
Q1-90
COMMITTED
ACCRUED
TARGET
(d)
AIM
(e)
CROMWELL
TOCCATA
HARPO
DM
(f)

FIG. 3

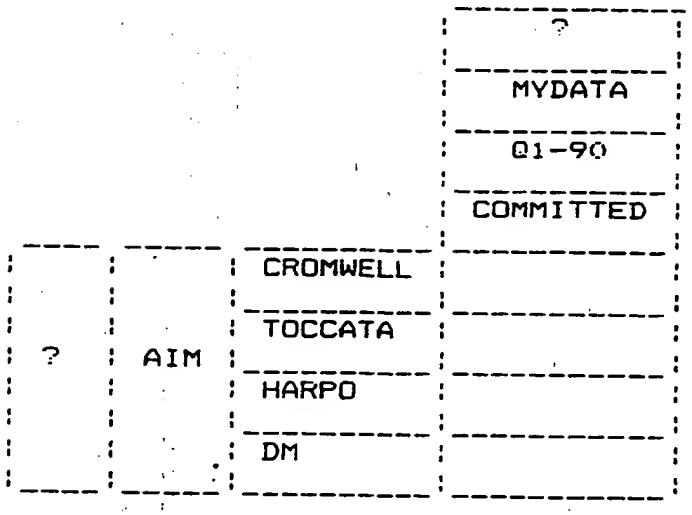
FIGURE 3(g)


?
MYDATA
Q1-90
?
AIM

(a)


?
MYDATA
Q1-90
COMMITTED
?
AIM

(b)


?
MYDATA
Q1-90
COMMITTED
CROMWELL
TOCCATA
HARPO
DM
?
AIM

(c)

FIG. 4

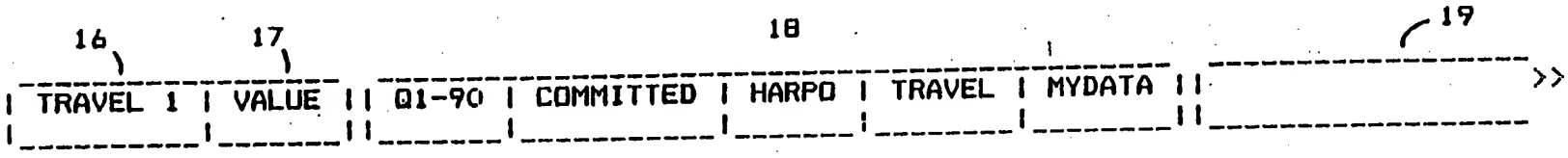

16
17
18
19
TRAVEL 1
VALUE
Q1-90
COMMITTED
HARPO
TRAVEL
MYDATA
>>

FIG. 5

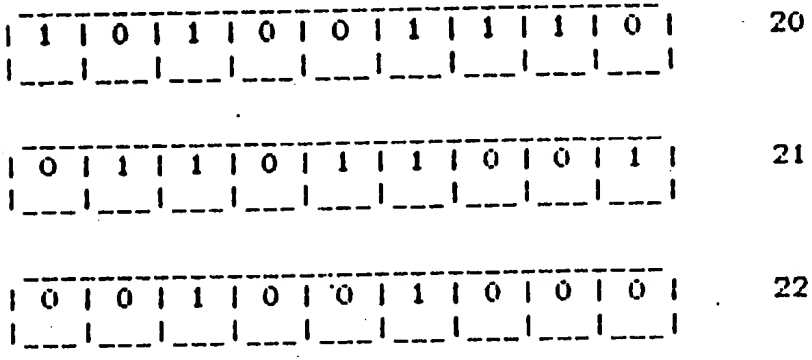

1 0 1 0 0 1 1 1 0
20
0 1 1 0 1 1 0 0 1
21
0 0 1 0 0 1 0 0 0
22

FIG. 6

	AIM
TRAVEL	64·7

FIGURE 7

AIM			
TRAVEL			
FY-91			
Q1-91	Q2-91	Q3-91	Q4-91
7·8	22·1	16·7	18·1

FIGURE 8

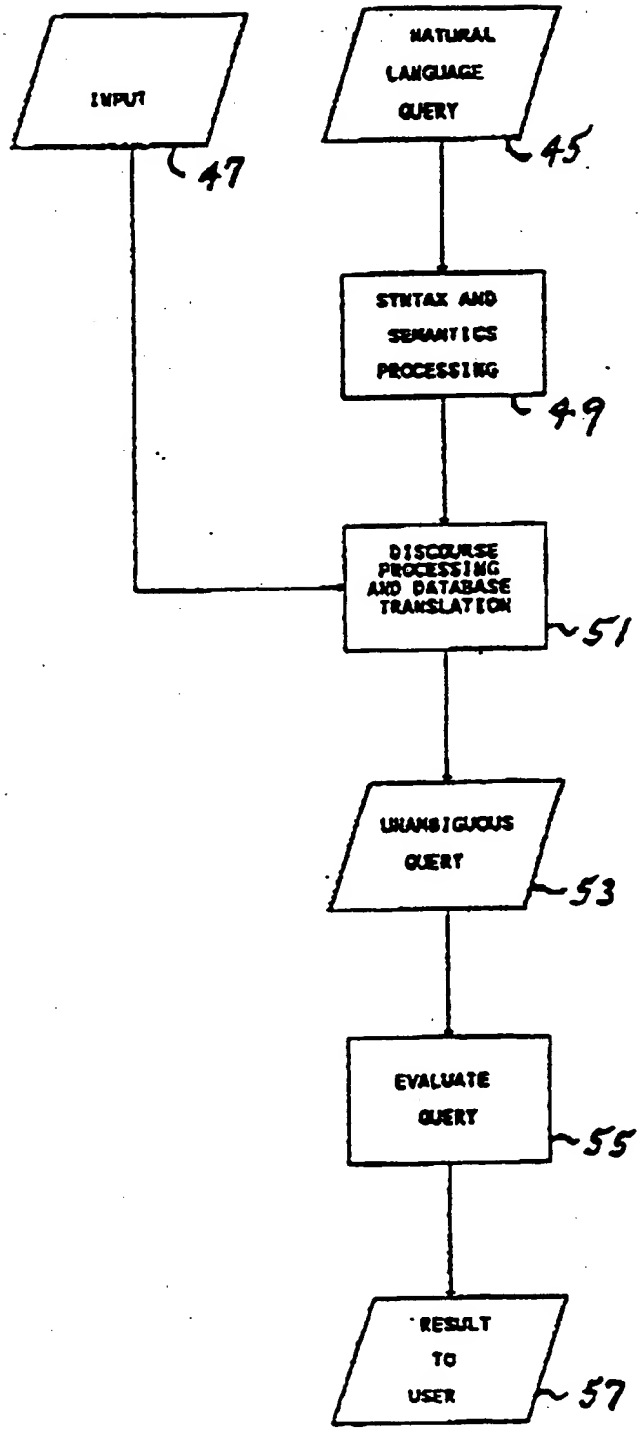
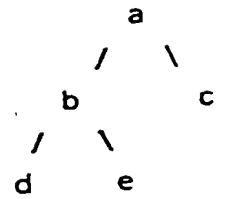
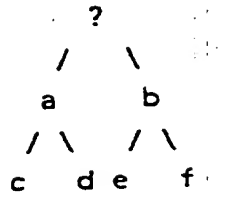

INPUT
47
NATURAL LANGUAGE QUERY
45
SYNTAX AND SEMANTICS PROCESSING
49
DISCOURSE PROCESSING AND DATABASE TRANSLATION
51
UNAMBIGUOUS QUERY
53
EVALUATE QUERY
55
RESULT TO USER
57

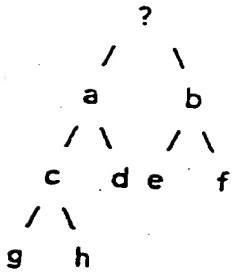
Fig. 9



(a)



(b)



(c)

FIG 10

INTERNATIONAL SEARCH REPORT

International Application No PCT/GB 91/00200

I. CLASSIFICATION OF SUBJECT MATTER (If several classification symbols apply, indicate all) [6]

According to International Patent Classification (IPC) or to both National Classification and IPC

IPC[5]: G 06 F 15/40

II. FIELDS SEARCHED

Minimum Documentation Searched [7]

Classification System	Classification Symbols
IPC[5]	G 06 F 15/40

Documentation Searched other than Minimum Documentation to the Extent that such Documents are included in the Fields Searched [8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT [9]

Category *	Citation of Document, [11] with indication, where appropriate, of the relevant passages [12]	Relevant to Claim No. [13]
X	ACM Transactions on Database Systems, vol. 14, no. 4, December 1989, ACM, (New York, US), G. Özsoyoglu et al.: "Query processing techniques in the summary-table-by-example database query language", pages 526-573 see abstract; page 527, line 36 - page 528, line 18; figure 1; page 532, line 34 - page 533, line 10 --	1-6,8-10
A	IEEE Computer Graphics and Applications, vol. 14, no. 5, May 1981, IEEE, M.M. Zloof: "QBE/OBE: A language for office and business automation", pages 13-21 see the whole article --	1-6,8-10
A	Software Practice & Experience, vol. 19, no. 6, June 1989, John Wiley & Sons, Ltd., (Chichester, Sussex, GB), ./.	1-6,8-10

* Special categories of cited documents: [10]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search	Date of Mailing of this International Search Report
14th May 1991	16 JUL 1991
International Searching Authority	Signature of Authorized Officer
EUROPEAN PATENT OFFICE	MISS I. TAZELAAR

III. DOCUMENTS CONSIDERED TO BE RELEVANT (CONTINUED FROM THE SECOND SHEET)

Category *	Citation of Document, ** with indication, where appropriate, of the relevant passages	Relevant to Claim No.
	L.A. Rowe et al.: "A visual shell interface to a database", pages 515-528 see the whole article	
	--	
A	IEEE Computer Software and Applications Conference, Compsac '87, 7-9 October 1987, Tokyo, JP, C.T. Wu: GLAD: Graphics language for database", pages 164-170 see the whole article	1-6,8-10

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☑ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

This Page Blank (uspto)